Apprentissage Automatique

Thomas JAILLON

I. Preprocessing

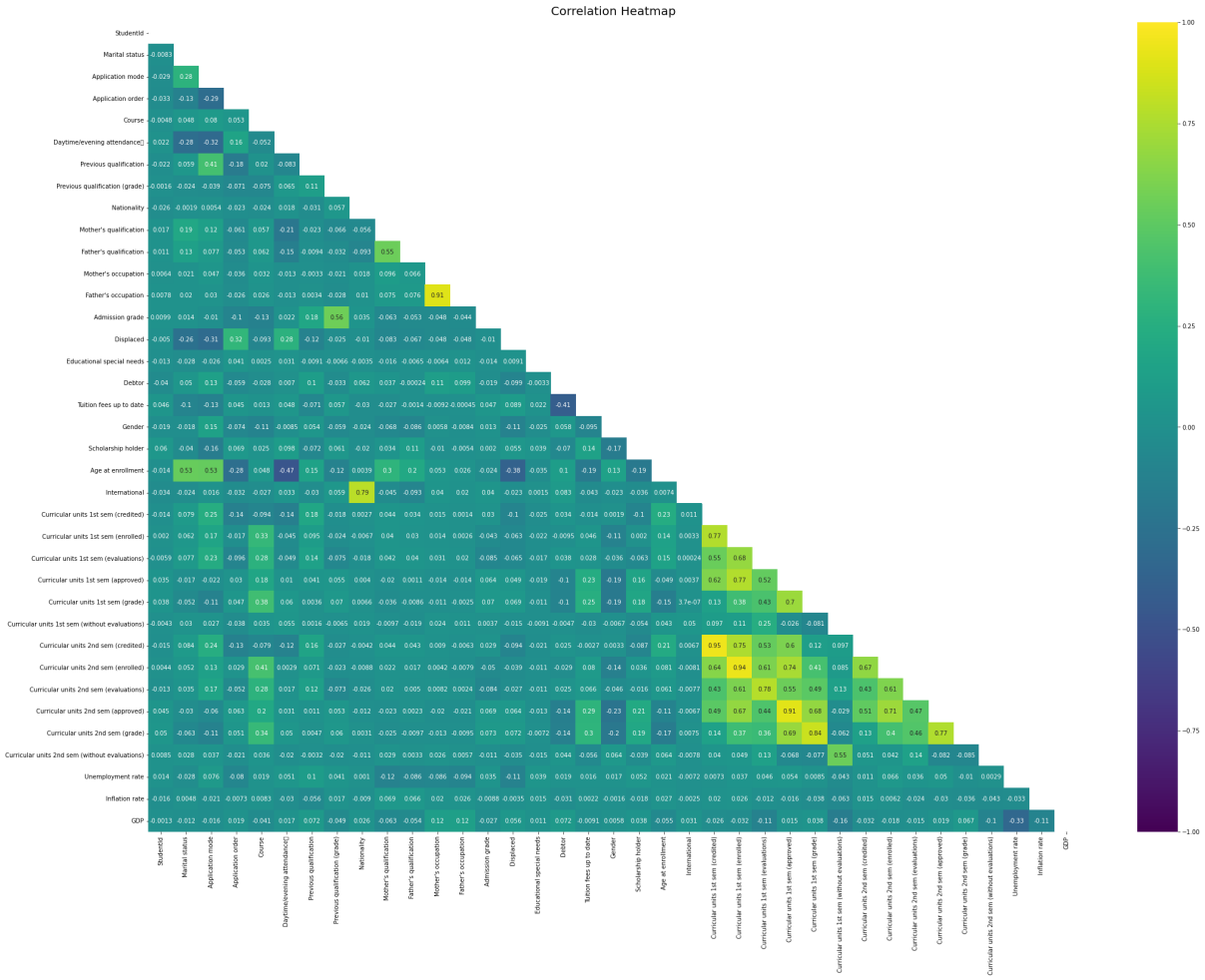
A. Présentation des données

Le jeu de données “Predict students' dropout and academic success” se divise en 37 colonnes. Il est déjà préparé dans la mesure où il a été encodé, ne comporte pas de valeurs aberrantes ni de valeurs manquantes. Pour analyser nos données j’ai utilisé les commandes suivantes:
df.columns pour récupérer les colonnes.
df.info() pour connaître le type et le nombre de données non nulles de chaque colonnes.
df.head() pour prévisualiser les premières lignes et donc données.
En les analysant grâce à leur type et description on peut établir quelles sont les variables catégorielles, ordinales, binaires ou numériques. J’ai donc pu établir les listes suivantes qui serviront ensuite lors du preprocessing:

```
Category_list = ["Marital status","Application mode","Course","Previous qualification","Nationality", "Mother's qualification", "Mother's occupation","Father's occupation","Father's occupation"]
Binary_list = ["Daytime/evening attendance\t","Displaced","Educational special needs","Debtor","Tuition fees up to date","Gender","Scholarship holder","International"]
Ordinal_list = ['Application order']
Numerical_list = ['Previous qualification (grade)','Admission grade','Age at enrollment','International','Curricular units 1st sem (credited)','Curricular units 1st sem (enrolled)','Curricular units 1st sem (evaluations)','Curricular units 1st sem (approved)','Curricular units 1st sem (grade)','Curricular units 1st sem (without evaluations)','Curricular units 2nd sem (credited)','Curricular units 2nd sem (enrolled)','Curricular units 2nd sem (evaluations)','Curricular units 2nd sem (approved)','Curricular units 2nd sem (grade)','Curricular units 2nd sem (without evaluations)','Unemployment rate','Inflation rate','GDP']
```

B. Features Engineering

A présent je vais approfondir l'analyse des données. Premièrement j'ai affiché la table de corrélations des différentes variables:

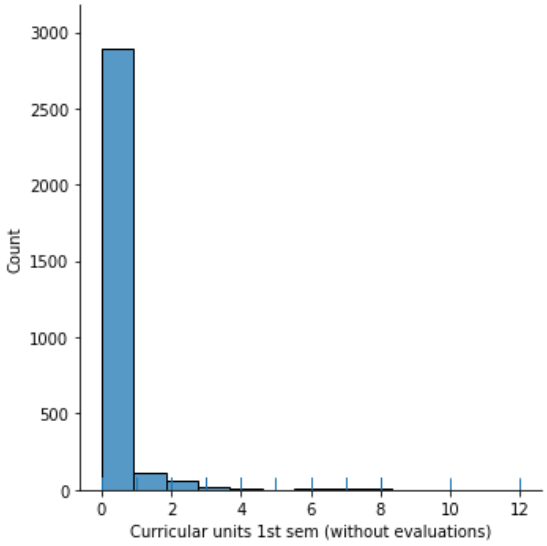


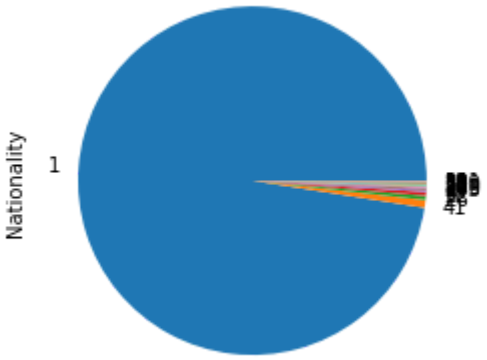
Certaines variables sont très corrélées avec une corrélation supérieure à 0,90 notamment les paires de variables suivantes:

- "Father's occupation" et "Mother's occupation" à 0,91
- "Curricular units 1st sem (approved)" et "Curricular units 2nd sem (approved)" à 0,91
- "Curricular units 1st sem (enrolled)" et "Curricular units 2nd sem (enrolled)" à 0.94
- "Curricular units 1st sem (credited)" et "Curricular units 2nd sem (credited)"

J'ai également utilisé les fonctions "displot" et "plot.pie" de la libraire SEABORN pour étudier la répartition des données au sein des variables. Il est apparu que de nombreuses variables ont une répartition très inégale des données comme on peut le voir ci-dessous et dès lors présente un faible intérêt dans la prédiction de la variable "Target". On peut notamment relever les variables suivantes: "Curricular units 1st sem (credited)", "Curricular units 2nd sem (credited)", "Curricular units 1st

sem (without evaluations)", "Curricular units 2nd sem (without evaluations), "Nationality" et "Educational special needs".





J'ai également défini des fonctions d'encodage, Ordinal, Category et OneHot encoding que l'on retrouve sur les notebook. Au cours des différents essais j'ai utilisé ces fonctions sur les différentes variables. J'ai aussi utilisé différentes normalisations comme la normalisation "MinMaxScaler". Cependant je n'ai pas retenu ce choix dans mes 3 rendus finaux car les résultats associés s'étaient révélés inférieurs.

Après avoir analysé les résultats de corrélations, d'influence et de répartitions de mes variables j'ai fait le choix de retirer les variables suivantes:

```
['Curricular units 1st sem (credited)', 'Curricular units 1st sem (without evaluations)', 'Curricular units 2nd sem (credited)','Curricular units 2nd sem (without evaluations)', 'Nationality','Educational special needs','International']
```

En effet, elles n'impactent que trop peu l'apprentissage des modèles et par conséquent surcharge inutilement l'opération.

II. Apprentissage

A. Choix des Algorithmes

Pour choisir les algorithme adéquats j'ai créé une liste "classifier" ou j'ai ajouté plusieurs algorithmes de classification: SVC, DecisionTreeClassifier, AdaBoostClassifier, RandomForestClassifier, ExtraTreesClassifier, GradientBoostingClassifier, MLPClassifier, KNeighboorsClassifier, LogisticRegression et enfin LinearDiscriminantAnalysis. Afin d'évaluer ces différents

modèles j'ai effectué une validation croisée avant d'afficher les résultats dans un tableau. La validation croisée permet de mesurer les performances des modèles avec leur paramètre dans un espace de données plus grand puisqu'il permet d'utiliser la totalité du jeu de données pour l'apprentissage et l'évaluation et non une partie seulement. Cela m'a permis d'identifier les modèles les plus performants rapidement à savoir les Random Forest et Logistic Regression.
Au cours de la compétition Kaggle j'ai également découvert et utilisé la librairie python Pycaret qui permet d'automatiser les flux de machine learning. Pycaret se charge de comparer les modèles et retourne le plus performant avec les paramètres optimisés.
J'ai également utilisé les modèles vus au cours de l'unité et donc choisi d'implémenter un modèle de Stacking. Les trois solutions que j'ai rendues sont donc un modèle de random forest, de logistic regression et un enfin de stacking. J'ai choisi la Random Forest car il s'agissait de mon meilleur score et j'ai choisi la régression logistique et le Stacking car il me semblait plus robuste.
Pour implémenter l'algorithme de stacking j'ai sélectionné les algorithmes les plus performants grâce à ma liste de classifiers évaluée par validation croisée décrite précédemment et j'ai obtenu le tableau suivant:

	CrossValMeans	CrossValerrors	Algorithm
1	0.672853	0.025922	DecisionTree
7	0.675279	0.022102	KNeighboors
2	0.679729	0.033266	AdaBoost
0	0.760095	0.012927	SVC
9	0.764139	0.022873	LinearDiscriminantAnalysis
6	0.770202	0.017142	MultipleLayerPerceptron
8	0.773012	0.012761	LogisticRegression
5	0.773830	0.020395	GradientBoosting
10	0.775441	0.019057	XGBClassifier
11	0.775851	0.013841	CatBoostClassifier
3	0.777865	0.014921	RandomForest
4	0.778678	0.013431	ExtraTrees

J'ai donc sélectionné l'ensemble des algorithmes avec une accuracy par validation croisée supérieure à 0,77. Puis j'ai construit mon algorithme de Stacking en listant ces différents algorithmes:

```
random_state=21
```

```
estimators = [
('mlp',MLPClassifier(random_state=random_state)),
('gb',GradientBoostingClassifier(random_state=random_state,ccp_alpha=0.
0,criterion='friedman_mse',init=None,learning_rate=0.1,
loss='deviance',max_depth=3,max_features=None,max_leaf_nodes=None,min_i
mpurity_decrease=0.0,min_samples_leaf=1,min_samples_split=2,min_weight_
fraction_leaf=0.0,n_estimators=100,n_iter_no_change=None,subsample=1.0,
tol=0.0001, validation_fraction=0.1,verbose=0,warm_start=False,)),
('xgb', xgb.XGBClassifier(random_state=random_state)),
('cb',CatBoostClassifier(random_state=random_state)),
('rf',RandomForestClassifier(random_state=random_state,max_depth=None,m
ax_features='auto',min_samples_leaf=1,min_samples_split=4,n_estimators=
1000)),
('et',ExtraTreesClassifier(random_state=random_state))]
clf = StackingClassifier(
       estimators=estimators, final_estimator=LogisticRegression(C=1.0,
class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=1000,
                   multi_class='auto', n_jobs=None, penalty='l2',
                         random_state=4695, solver='lbfgs', tol=0.0001,
verbose=0,
                   warm_start=False)
)
```

III. Evaluation

A. Choix des paramètres

Pour optimiser les paramètres des différents algorithmes j'ai principalement utilisé les fonctions gridsearchCV et RandomizedSearchCV. Il faut définir une grille comportant l'ensemble des possibilités des différents paramètres que l'on souhaite tester. On exécute ensuite l'ensemble des possibilités et on obtient la meilleure combinaison.
Exemple de grilles de paramètres:
Pour un RandomForest et une LogisticRegression:

```
grid = {"n_estimators": [10, 100, 200, 500, 1000, 1200],
"max_depth": [None, 5, 10, 20, 30], "max_features": ["auto", "sqrt"],
"min_samples_split": [2,4,6], "min_samples_leaf": [1, 2, 4]}

param_grid = {'penalty': ('l1', 'l2'), 'C': [0.1, 0.5, 1.0, 1.5, 2.0],
'class_weight': ('balanced', None),'max_iter':[1000,10000,50000]}
```

Grâce à ces méthodes de recherche j'ai retenu les paramètres suivants pour ma régression logistique et Random Forest:

```
LogisticRegression(C=0.1,max_iter=50000,penalty='l1',solver='liblinear'
)
RandomForestClassifier(bootstrap=False,max_depth=20,max_features='auto'
,min_samples_leaf=2,min_samples_split=2,n_estimators=1800)
```

Pour les paramètres de stacking j'ai réutilisé les meilleurs paramètres des algorithmes que j'avais déjà pu calculer au cours de mes différents essais ou bien laisser le paramétrage par défaut.

B. Performance

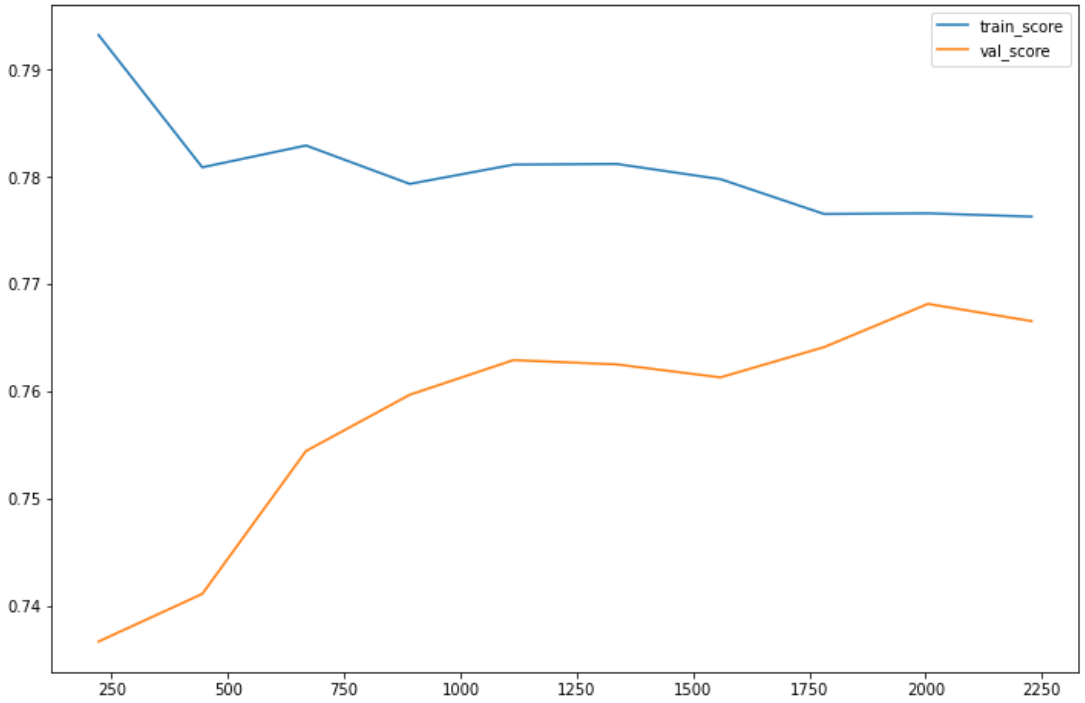
Pour améliorer l'analyse des résultats j'ai utilisé les matrices de confusions ainsi que les courbes d'apprentissage pour mieux contrôler la performance et l'overfitting. Le paramètre précision désigne le nombre de prédictions positives correctement effectuées sur l'ensemble des prédictions positives effectuées. Le recall désigne le nombre de résultats positifs correctement prédits par notre modèle sur l'ensemble des résultats positifs. Autrement dit plus la précision est haute moins le modèles se trompe sur les résultats positifs et plus le recall est grand plus le modèle identifie des positifs. Le paramètre F1-score, métrique d'évaluation du kaggle décrit le rapport entre précision et recall selon la formule suivante: $F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. Par exemple pour la régression logistique j'ai obtenu les résultats suivants:

```
[[132  17  32]
 [ 34  24  62]
 [  5   9 305]]
              precision    recall  f1-score   support

     Dropout       0.77      0.73      0.75       181
    Enrolled       0.48      0.20      0.28       120
    Graduate       0.76      0.96      0.85       319

    accuracy                           0.74       620
   macro avg       0.67      0.63      0.63       620
weighted avg       0.71      0.74      0.71       620

accuracy 0.7435483870967742
```



J’ai choisi de rendre ce modèle malgré le fait qu’il ne possédait pas la meilleure accuracy car il me semblait robuste notamment parce qu'il ne fait pas d’overfitting.

Pour la Random Forest j’ai obtenu les résultats suivants:

```
[[212  25  54]
 [ 41  54  68]
 [ 14  19 442]]
              precision    recall  f1-score   support

     Dropout       0.79      0.73      0.76       291
    Enrolled       0.55      0.33      0.41       163
    Graduate       0.78      0.93      0.85       475

    accuracy                           0.76       929
   macro avg       0.71      0.66      0.67       929
weighted avg       0.75      0.76      0.75       929

accuracy 0.7621097954790097
```

Le modèle Random Forest possède de bons résultats mais est en overfitting.

Pour l’algorithme de stacking j’ai obtenu les résultats suivants

```
[[131  26  24]
 [ 26  53  41]
 [  9  29 281]]
              precision    recall  f1-score   support

     Dropout       0.79      0.72      0.76       181
    Enrolled       0.49      0.44      0.46       120
    Graduate       0.81      0.88      0.85       319

    accuracy                           0.75       620
   macro avg       0.70      0.68      0.69       620
weighted avg       0.74      0.75      0.75       620

accuracy 0.75
```

J’ai rendu l’algorithme de stacking car il est selon moi le plus robuste de l’ensemble de mes modèles. En effet, grâce à un estimateur final, une régression logistique, il permet de combiner les sorties, prédictions de plusieurs autres modèles individuels. Les performances du Stacking sont généralement proches du meilleur modèle et peuvent parfois dépasser les performances de prédiction de chaque modèle individuel.

Conclusion:

Au cours de la compétition kaggle j’ai réalisé de nombreux modèles différents. J’ai rencontré plusieurs cas d’overfitting notamment avec les Random Forest. J’ai tout de même rendu un modèle implémentant une Random Forest car il possédait une grande précision. Grâce à l’analyse des données j’ai pu effectuer plusieurs opérations de preprocessing comme: l’élimination de variables, l’encodage ou la normalisation. J’ai évalué de nombreux algorithmes par validation croisée et j’ai implémenté les plus performants. J’ai aussi combiné ces algorithmes dans un modèle de stacking plus robuste pour pouvoir être performant sur l’évaluation finale. J’ai cherché à optimiser les paramètres des différents modèles grâce aux méthodes de test de grilles de paramètres (GridSearchCV et RandomizedSearchCV). Enfin j’ai évalué la performance de mes modèles sur plusieurs critères comme la précision, le recall, le F1-score ou encore en analysant les courbes d’apprentissage et d’évaluation.